Proceedings of the 7th International Conference on HydroScience and Engineering

Philadelphia, USA September 10-13, 2006 (ICHE 2006)

ISBN: 0977447405

Drexel University

College of Engineering

HIGH PERFORMANCE COMPUTING IN HYDRO- AND ENVIRONMENTAL ENGINEERING

Reinhard Hinkelmann[1]

ABSTRACT

High Performance Computing (HPC) can be understood as the interaction of parallel and adaptive methods with fast solvers on powerful parallel computers. Therefore, an introduction to parallel and adaptive methods as well as to fast solvers is given. The interaction of these methods is demonstrated using three examples which deal with groundwater flow and transport processes, gas-water flow as well as multiphase / multicomponent flow and transport processes in the subsurface. HPC has become applicable for a number of problem classes and modeling systems in hydro- and environmental sciences with a reasonable effort of adapting or further developing source codes. HPC opens new horizons for simulating and understanding complex coupled processes in large scale systems.

1. INTRODUCTION

The tremendous ongoing increase in computer technology in the last two decades has led to a change in methods applied in hydro- and environmental sciences and engineering. In addition to theory and experiment, *High Performance Computing* (HPC) based on mathematical models, computer simulation and optimization has become the third pillar of research. HPC offers new ways of broadening the knowledge of complex coupled processes and phenomena in research and of addressing large space and time scales in hydro- and environmental engineering practice. There is no doubt that a number of tomorrow's fundamental problems will be solved with its help.

On the hardware side, HPC is dominated by parallel computer systems consisting of scalar processors or vector units. On the software side, several 'general purpose' tool-boxes have been developed in recent years which allow the migration of a large number of source codes to parallel systems with comparatively small number of adaptations, i.e. with little effort. In such cases, no further developments in parallelization have to be carried out for a specific source code. The above-mentioned tool-boxes often use efficient numerical methods such as fast solvers (CG and Multigrid solvers) and adaptive methods. Consequently, HPC can be understood as the interaction of parallel and adaptive methods as well as fast solvers on a suitable hardware platform (see Fig. 1). Detailed information on these topics is found, for example, in Bastian (1996) and Hinkelmann (2005).

The paper is organized as follows: In section 2, parallel methods and hardware aspects are discussed. Adaptive methods are introduced in section 3, while section 4 deals with fast solvers. In

[1] Professor, Department of Water Resources Management and Hydroinformatics, Technische Universität Berlin, Germany (reinhard.hinkelmann @wahyd.tu-berlin.de)

section 5, a list of HPC toolboxes is given. Three examples are presented in section 6, and conclusions are drawn in section 7.

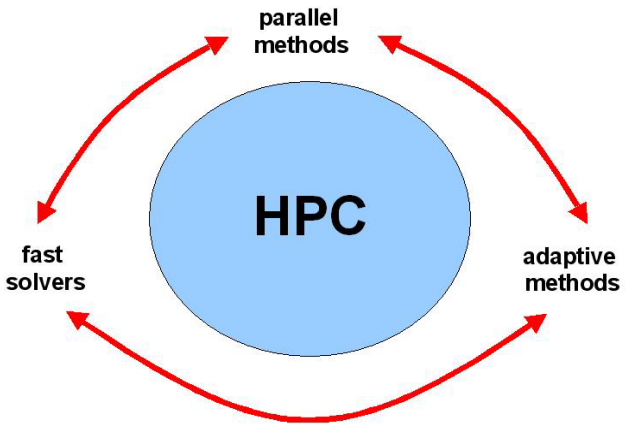

Figure 1 Fundamentals of HPC.

2. PARALLEL METHODS

Parallel computing has opened new dimensions in solving large space and time scale problems because hundreds or thousands of processors can be connected for their solution. In most cases, the *Multiple Instruction Multiple Data* (MIMD) principle has proven to be the best one for parallel computing. Here, a small or large number of autonomous processors can operate fully independently of each other, and the processors are connected by a fast communication network which is called functional parallelism. The *Message-Passing Interface* (MPI, http://mpi-forum.org) has developed to be a quasi-standard for the communication which is carried out with the very fast *Myrinet* (http://www.myri.com/myrinet) or with the slower *Fast Ethernet* from the hardware side. The fastest computers worldwide are ranked in the list *TOP500 Supercomputing Sites* (http://www.top500.org). At the moment, this list is clearly dominated by *clusters* and *Massively Parallel-Processing* (MPP) systems. MPPs consist of a large number of nodes where each node is equipped with one processor, while clusters have a large number of nodes which consist of up to 4 processors each.

The *algebraic parallelization* strategy is the most widely used one because the whole or the major part of the serial algorithm remains the same. The parallelization is carried out on the level of basic algebraic operations such as matrix-vector product or vector operations. Every processor computes its part of the basic algebraic task with its assigned data. Many 'higher level' parts of an algorithm, for example the solver (see sec. 4), 'just' use these algebraic operations and do not have to consider the parallelization. However, some parts of an algorithm may need a special treatment for parallelization. Another parallelization strategy is explained in section 6.3; it is much simpler, however just applicable for some special cases.

The computational load must be divided among the processors in such a way that they are equally burdened and that the interprocessor communiction is minimized. The *load balancing* is done by partitioning the computational domain into subdomains and assigning each subdomain together with the corresponding data to one processor (see Fig. 2, right). If the computational load increases significantly in subdomains, for example caused by adaptive mesh refinement (see sec. 3), a load redistribution should be carried out during run time; this is called *dynamic load balancing*.

The *parallel speedup* is a definition for the performance quality of a parallel algorithm. It is defined as the ratio of the run time on 1 processor to the run time on p processors. Generally, the parallel speedup increases with increasing number of processors (see Fig. 2, left).

It is referred to section 5 where HPC toolboxes which support parallel computing are listed. Further reading on parallel computing is given, for example, in Bastian (1996) and Birken (1998).

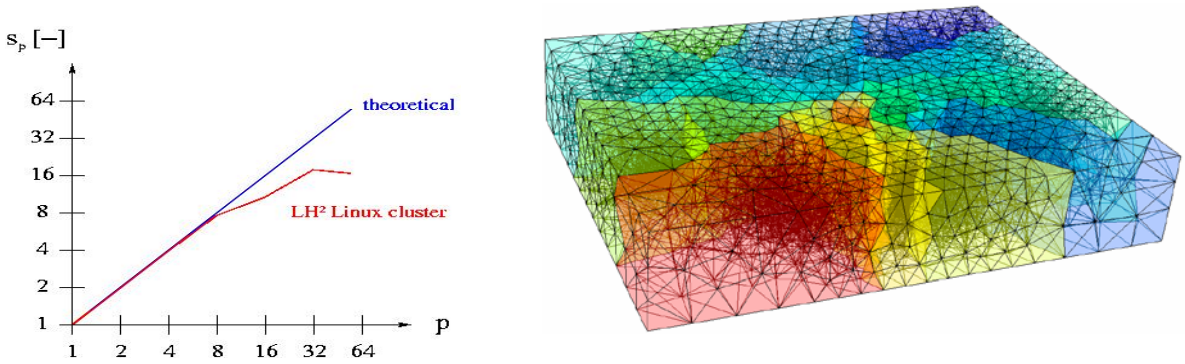

Figure 2 Parallel speedup (left) and distributed grid (right).

3. ADAPTIVE METHODS

Adaptive methods aim at numerical solutions with high accuracy, optimal computational effort and storage requirement by automatically adjusting the mesh or the solution method to the temporally and spatially variable solution function. First, *a priori* and *a posteriori* methods are distinguished depending on whether they are applied before or after a computation. In areas where it is generally known that the solution accuracy is poor or where sharp gradients of the solution can be expected, for example around wells or fractures in subsurface flow, an a priori mesh refinement can be carried out (see Fig. 3). A posteriori methods are discussed in the following.

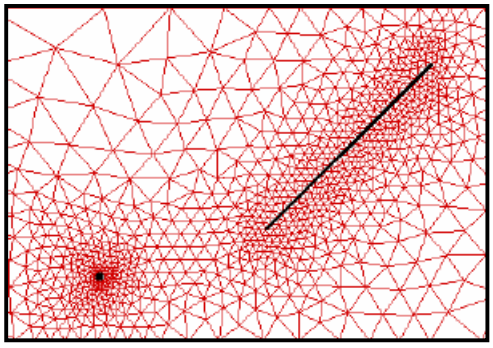

Figure 3 A priori mesh refinement.

Among different methods and techniques of adaptation, the *h-adaptive method* in which the mesh is refined or coarsened controlled by some error criteria is most widely used. These error criteria consist of heuristic *error indicators* or mathematically proven *error estimators*. The indicators detect sharp fronts by means of differences, gradients or curvatures of the solution function as shown in Figure 4 for a sharp front of the water saturation. Estimators are only applied in a limited number of problems, for example groundwater flow. They compute an error in the energy norm based on jumps of computed variables along element edges, for example jumps of Darcy velocity. Sometimes, adaptations in time are carried out, for example by adjusting a time step to a stability criterion in an explicit method or to an iteration number of a linear or non-linear solver (see sec. 4) in an implicit method. For the *refinement* and *coarsening*, tolerances must be defined which relate an error in an element, for example to an average error or a maximum error of all elements.

During the mesh coarsening, the strategies must ensure the conservativity of the involved balance equations, for example mass and momentum conservativity.

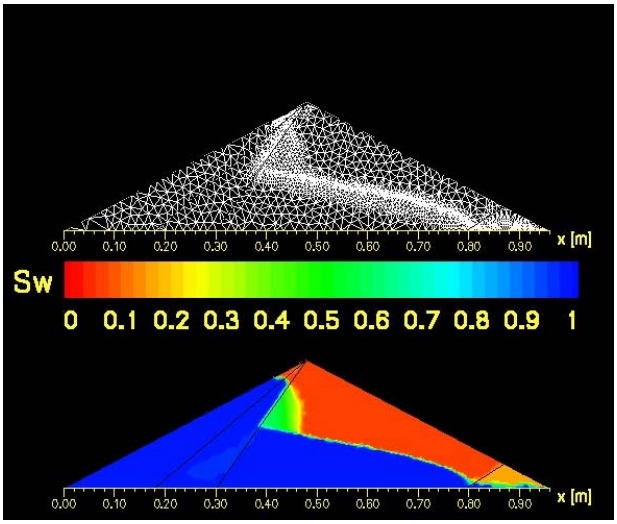

Figure 4 A posteriori mesh refinement.

It is refered to section 5 where HPC toolboxes which support adaptivity are given. Further information on adaptive methods is found, for example, in Johnson (1990) and Verfürth (1996).

4. FAST SOLVERS

Most discetization methods are implicit and lead to the solution of large systems of unknowns. As the solver generally requires the major part of the CPU time, it is of particular importance and must be considered in the context of parallel and adaptive methods (see sec. 2, 3). First, *linear* and *non-linear solvers* are distinguished depending on whether the coefficients in the system matrix are constant or depend on the unknowns. In the following, we discuss linear solvers and we come back to non-linear solvers at the end of this section.

We differentiate between *direct* and *iterative solvers*. Direct methods are generally based on a successive elimination of the unknowns with the *Cholesky* or *Gauss algorithm.* Iterative methods determine the solution as a limit of a series of approximations which is terminated if a stopping criterion is fulfilled. In general, so-called *sparse matrices* are dealt with in hydro- and environmental engineering. Sparse means that there are only a few non-zero entries in every line and column of the system matrix. Special storage techniques which just address the non-zero terms have been developed. As the computational effort for directly solving sparse matrices is in the order $O(n^2)$ with n being the number of unknowns, direct methods are not suitable for HPC as the only solution method.

Iterative methods use a single or multiple grids to determine the solution. Single-grid methods just require the computational grid, and they are faster than direct solvers. The classical iterative solvers are those of *Jacobi* and *Gauss-Seidel*. However, *Conjugate Gradient* (CG) methods have become superior in the last two decades (see Fig. 5, left). In this context, the *Preconditioned Conjugate Gradient* (PCG) method has developed to be the ‘standard solver’ for symmetric matrices, while the *Biconjugate Gradient Stabilized* (BiCGSTAB) method as well as the *Generalized Minimal Residual* (GMRES) method have become the ‘standard solvers’ for non-symmetric matrices. All of the above mentioned methods can often be accelerated with

preconditioners which improve the condition of the matrix with some computationally cheap measures. The suitability of preconditioners for parallel computing (see sec. 2) must be considered. The later explained Multigrid methods can also be used for preconditioning. The computational effort for solving sparse matrices with Conjugate Gradient methods can be reduced up to the order $O(n^{1.5})$. They are recommended for medium-size systems with several thousands to several hundred thousands of unknowns.

Multigrid (MG) methods need several generally nested grids for the iteration process (see Fig. 5, right). Therefore, they need solvers on the coarse and fine grids as well as transfer operations between the grids. The classical coarse grid solver is a direct one, and for the fine grids, the Jacobi or Gauss-Seidel methods are often chosen. MG methods operate in an additive or multiplicative way. The computational effort for solving sparse matrices can be reduced up to $O(n^1)$. They are recommended for large-scale problems with more than hundred thousands or millions of unknowns. Although some components of a MG algorithm might not be optimal for parallel computing (see sec. 2), very good parallel performances are often achieved for the whole MG algorithm. Of course, iterative methods can be well combined with adaptive methods (see sec. 3), especially MG methods (see Fig. 5, right) because they use similar data structures.

For non-linear systems, the *Picard* and *Newton-Raphson* methods can be chosen. The Picard solver has a linear convergence behavior, while the Newton-Raphson has a quadratic one. Therefore, generally the Newton-Raphson method is superior. However, if the system is strongly non-linear, the Picard method can be better.

It is refered to the next section where HPC toolboxes which include fast solvers are listed. Further reading on fast solvers is given in Briggs et al. (1999) and Meister (1999).

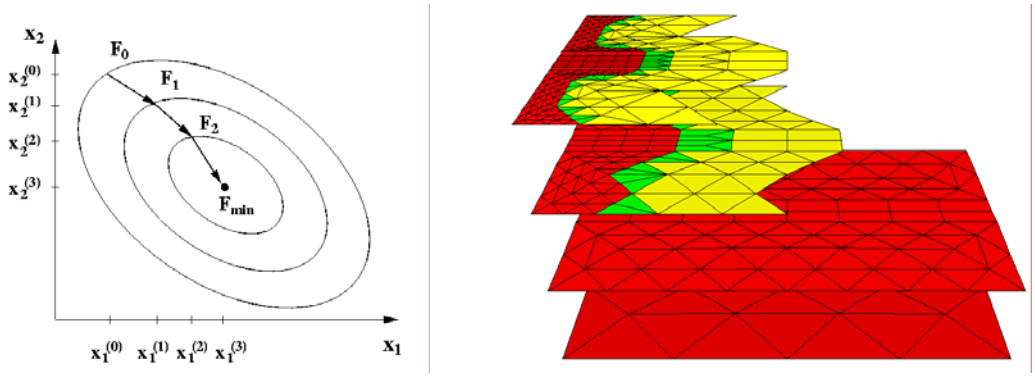

Figure 5 Conjugate Gradient (left) and adaptive Multigrid method (right).

5. HPC TOOLBOXES

As already mentioned in the previous sections, there are a number of ‘general purpose’ HPC toolboxes which include parallel and/or adaptive and/or fast solvers. These toolboxes should be used if one wants to migrate a source code to HPC systems because in many cases only some adaptations must be carried out and no or only little further developments are required. Thus, the effort of migration is comparatively small and the benefits of HPC can be huge. In the following, these toolboxes are just listed in an alphabetical order:

- **CHACO,** http://www.sandia.gov/CRF/chaco.html
- **COVISE,** http://www.hrls.de/organization/vis/covise
- **EXDASY,** http://www.gup.uni-linz.ac.at/research/exdasy

- **PADFEM,** http://math-www.uni-paderborn.de/cgi-bin/frame/mathepb
- **ParMETIS,** http://www-users.cs.umn.edu/karypis/metis
- **ScaLAPACK,** http://www.netlib.org/scalapack
- **Sumaa3D,** http://www-unix.mcs.anl.gov/sumaa3d
- **TEMPLATES,** ftp://ftpnetlib2.cs.utk.edu
- **UG,** http://cox.iwr.uni-heidelberg.de/ug/index.html

6. EXAMPLES

The following simulations are carried out with the numerical subsurface-simulator MUFTE-UG (see Helmig (1997), Bastian (1996), Hinkelmann (2005)). Mufte stands for multiphase flow, transport and energy model, while UG is the abbreviation of unstructured grid. UG is a toolbox which provides data structures and fast solvers based on parallel adaptive multigrid methods (see sec. 5).

6.1 Adaptive Multigrid Method for Groundwater Flow and Transport Processes

In a fractured-porous media system which represents a vertical cut through a sandstone outcrop, flow and transport processes are investigated (see Fig. 6). Fractures are discretely taken into account using the combined model approach, i.e. fractures are one-dimensional elements embedded in a two-dimensional matrix. The system is closed on the top and bottom. Prescribed boundary conditions on the left and right boundary lead to a horizontal flow from left to right. A constant concentration is imposed along the left boundary.

In Figure 7 the concentration distribution after some time is shown. The concentration flows ‘fast’ through the fractures and ‘slowly’ through the matrix. Along the sharp concentration gradients, the grid is adaptively refined, i.e. in parts of the fractures and at the left boundary. In Figure 5, right, the so-called Local Multigrid Method (see Bastian (1996)) is shown for a very similar example. The first grid level has been uniformly refined, while the second and third ones are adaptiveley refined. Therefore, only the refined elements and some transitional elements are stored on the second and third grid level. Overall, the adaptively refined mesh requires about 20% of the CPU time and storage compared to a uniformly refined solution, i.e. achieving the same accuracy of the solution.

Detailed information is found in Ochs et al. (2002) and Hinkelmann (2005).

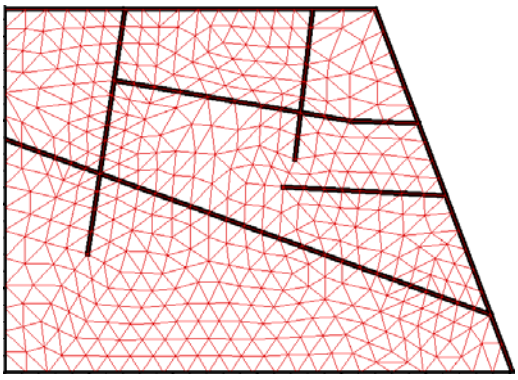

Figure 6 Fractured-porous system of a section through a sandstone outcrop

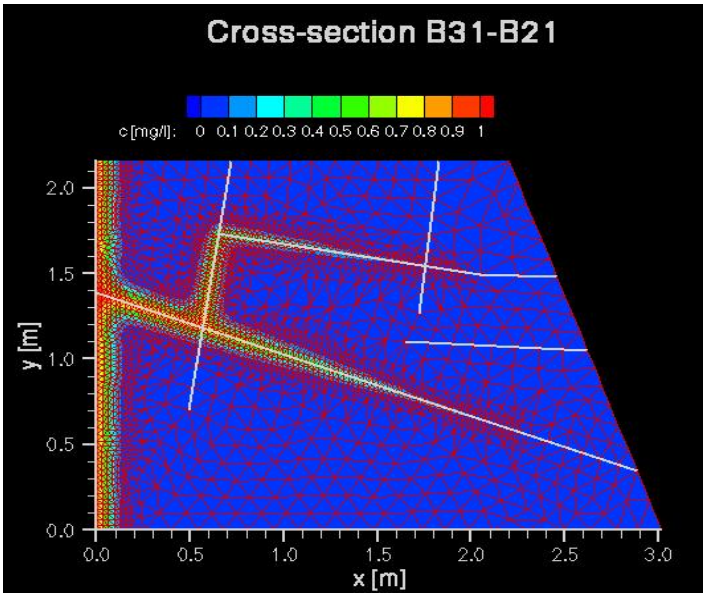

Figure 7 Concentration on adaptively refined grid.

6.2 Parallel Multigrid Method for Gas-Water Flow Processes in the Subsurface

In Figure 8 a three-dimensional subsurface system of an abandoned coal mine is presented. Coal seams are considered as two-dimensional elements, and a shaft as well as transport roads are discretized as one-dimensional structures. To investigate the influence of uncertainties in the geological structures, a fracture network was geostatistically generated. The coal seams are mainly directed in the horizontal and the fractures in the vertical direction. The distributed grid is shown in Figure 2, right. The different colors indicate different subdomains. The system is closed on all boundaries, only the top boundary is open. Initially, the system is fully filled with water. Gas (methane)-water flow processes are investigated. The system is driven by methane source terms which are located in the coal seams where the coal has not been exploited.

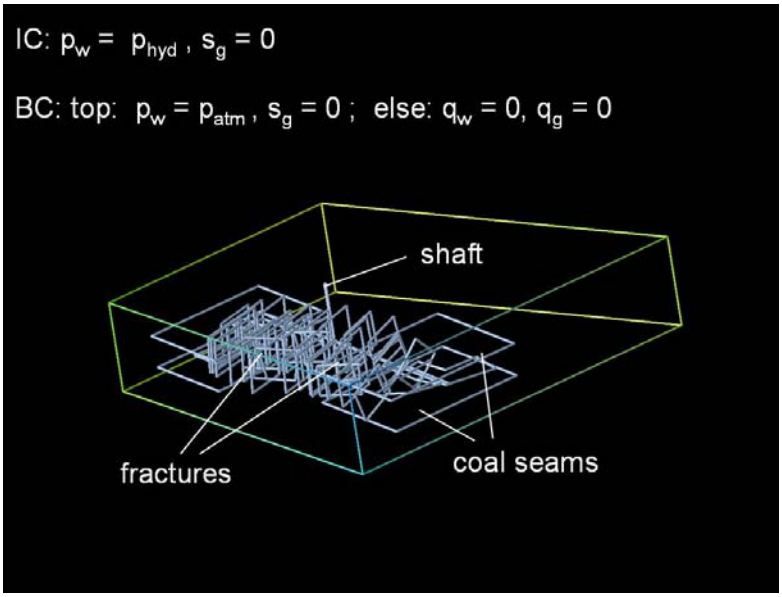

Figure 8 Fractured-porous system of a coal mine.

In Figure 9 the gas saturation after some time is shown. The methane migrates fast upwards through the fractures and then enters the matrix. It could be demonstrated that the geological structures have a significant influence on the amount of methane escaping to the atmosphere as well as its locations at the ground surface. The simulations were carried out with a parallel multigrid method (3 levels). In Figure 2, left, the parallel speedup obtained on a Linux cluster is given for this example. It indicates a reasonable performance up to 32 processors. However, the problem size with about 100000 elements was too small for more than 32 processors; this circumstance leads to a decrease in the inclination of the parallel speedup. We can conclude that the maximum number of processors must be adjusted to the problem size.

Detailed information is given in Sheta et al. (2003) and Breiting et al. (2004).

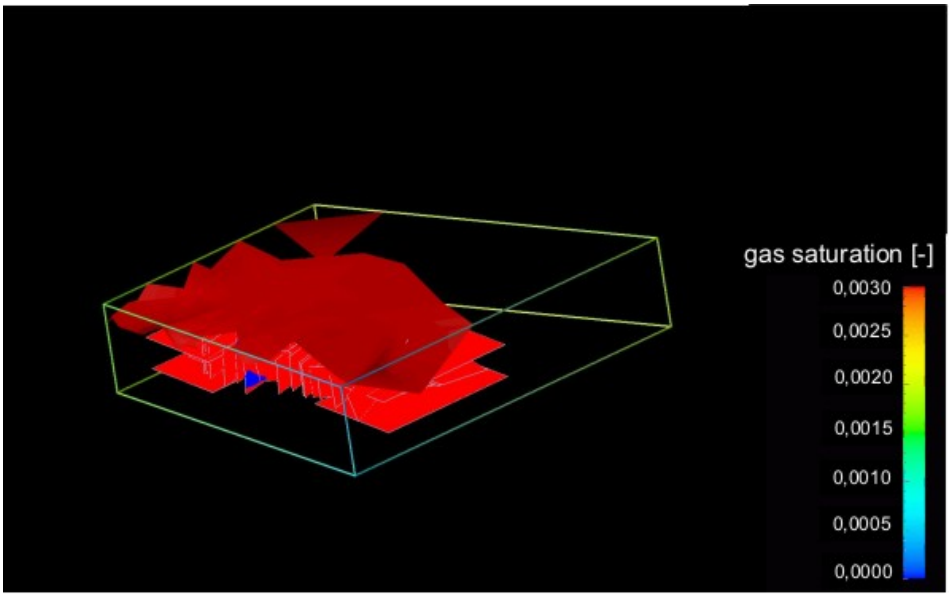

Figure 9 Gas saturation.

6.3 Parallel Optimization Method for Multiphase/Multicomponent Flow and Transport Processes in the Subsurface

Optimization methods have been developed to determine optimal locations and numbers for gas-extraction wells in order to keep buildings free of methane. The objective function to be minimized was the amount of extracted methane for a given underpressure in the well(s) (see Fig. 10, left) and a given time period. Simulated annealing has been implemented for the optimization. The methane is released from coal seams which are not exploited.

A system is investigated where methane flows from the coal seams to the transport roads which are partially filled with rock material and which have a high permeability. As a first step, we just considered the transport roads (see Fig. 10, right), and the methane inflow is prescribed by boundary conditions. This system is located in a domain being partially filled with water. We consider two fluid phases (water, gas) and three components in each phase (water (vapor), (dissolved) methane, (dissolved) air). The well(s) suck the gas, however, the extracted mass of methane should be minimized.

A simple *coarse-grain parallelization* strategy (see sec. 2) has been developed. The domain is divided into subdomains and each subdomain is assigned to a processor (see Fig. 10, right). The optimization method just searches the optimum in the subdomain, while the two-phase / three-component model is running in the whole domain (not in parallel). Therefore, the computations can

fully run in parallel, and at the end, the global optimum is determined by comparing the local optima in the subdomains. It is mentioned that the mesh must be newly generated in each iteration step of the optimization method because the well position is changing in each iteration step (see Fig. 10, left). Although 23 processors of a Linux cluster could be used (see Fig. 10, right), each simulation took several days, i.e. such simulations cannot be carried out without parallel computing.

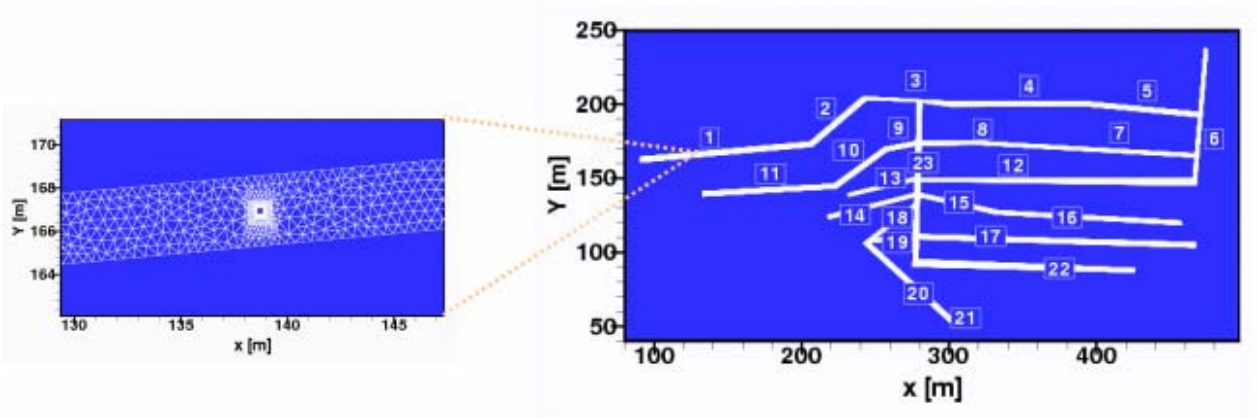

Figure 10 Zoom in subdomain 1 with the location of the well (left) and subdomains (right).

In Figure 11, right, the gas pressure is shown for a case with one well. The results represent the (local) optimum location of this well in subdomain 23. In Figure 11, left, the (local) optimum solutions of the methane gas extractions are given for each subdomain / segment. We see that the local optimum in subdomain 23 is the global optimum. However, the results in subdomains 13-15 are very similar. For the one-well case, the optimal locations are in the subdomains with crossings of the transport roads.

Detailed information is found in Kobayashi (2004) and Breiting et al. (2004).

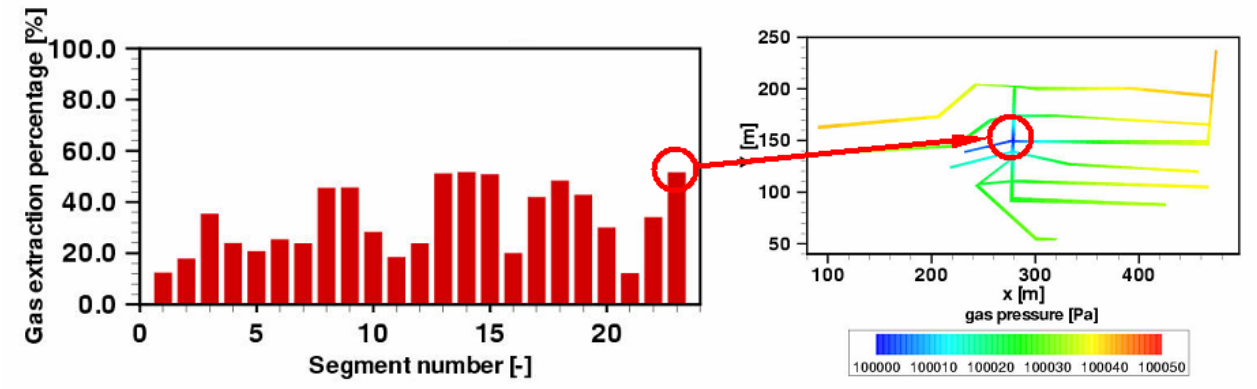

Figure 11 Gas extraction (left) and gas pressure in subdomain 23 (right).

7. CONCLUSIONS

High Performance Computing consists of the interaction of parallel and adaptive methods together with fast solvers on powerful parallel computers. It has become applicable for many problem classes and modeling systems in hydro- and environmental sciences with a reasonable effort of adapting or further developing source codes.

HPC is urgently required for modeling of hydrosystems on large space and time scales. It should be the basis of coupling models and domains as well as for upscaling, and it must be further developed in this context.

REFERENCES

Bastian, P. (1996). Parallele adaptive Mehrgitterverfahren, Teubner Skripte zur Numerik, B.G. Teubner, Stuttgart.

Breiting, T., Sheta, H., Kobayashi, K., Hinkelmann, R. and Helmig, R. (2004): Assessment of Hazardous Gas Emisson to the Surface over Fomer Mined Areas, Technischer Bericht Nr. 11/2004, Institut für Wasserbau, Universität Stuttgart.

Briggs, W.L., Hensen, V.E. and McCormic, S. (1999): A Multigrid Tutorial, 2nd edition, SIAM, Philadelphia.

Birken, K. (1998). Ein Modell zur effizienten Parallelisierung von Algorithmen auf komplexen, dynamischen Datenstrukturen, Dissertation, Universität Stuttgart.

Helmig, R. (1997). Multiphase Flow and Transport Processes in the Subsurface – A Contribution to the Modeling of Hydrosystems, Springer, Berlin Heidelberg New York.

Hinkelmann, R. (2005). Efficient Numerical Methods and Information-Processing Techniques for Modeling Hydro- and Environmental Systems, Springer, Berlin Heidelberg New York.

Johnson, C. (1990). "Adaptive Finite Element Methods for Diffusion and Convection Problems", Computer Methods in Applied Mechanics and Engineering 82, pp. 301-322.

Kobayashi, K. (2004). Optimization Methods for Multiphase Systems in the Subsurface – Application to Methane Migration in Coal Mining Areas, Dissertation, Mitteilungen Nr. 139, Institut für Wasserbau, Universität Stuttgart.

Meister, A. (1999). Numerik linearer Gleichungssysteme – Eine Einführung in moderne Verfahren, Vieweg, Braunschweig, Wiesbaden.

Ochs, S., Hinkelmann, R., Neunhäuserer, L., Helmig, R., Gebauer, S. and Kornhuber, R. (2002): "Adaptive Methods for the Equidimensional Modeling of Flow and Transport Processes in Fractured Aquifers", 5th International Conference on Hydrosciences & Engineering, Warsaw, Poland.

Sheta, H., Helmig, R. and Hinkelmann, R. (2003). "Dreidimensionale numerische Modellierung von Methanmigrationsprozessen im Untergrund", Proceedings Oberhausener Grubengastage 2003, Frauenhofer IRB Verlag, UMSCHICHT-Schriftenreihe 144, Stuttgart, pp. 135-140.

Verfürth, R. (1996). A Review of A Posteriori Error Estimation and Adaptive Mesh Refinement Techniques, John Wiley and Sons & B.G. Teubner Publishers, Stuttgart.